Contents

Canary Upgrades

6 minute read page test

Upgrading Istio can be done by first running a canary deployment of the new control plane, allowing you to monitor the effect of the upgrade with a small percentage of the workloads before migrating all of the traffic to the new version. This is much safer than doing an in-place upgrade and is the recommended upgrade method.

When installing Istio, the `revision` installation setting can be used to deploy multiple independent control planes at the same time. A canary version of an upgrade can be started by installing the new Istio version's control plane next to the old one, using a different `revision` setting. Each revision is a full Istio control plane implementation with its own `Deployment`, `Service`, etc.

Before you upgrade

Before upgrading Istio, it is recommended to run the `istioctl x precheck` command to make sure the upgrade is compatible with your environment.

```
$ istioctl x precheck
✔ No issues found when checking the cluster. Istio is safe to in
stall or upgrade!
  To get started, check out https://istio.io/latest/docs/setup/g
etting-started/
```

When using revision-based upgrades jumping across two patch versions is supported (e.g. upgrading directly from version `1.8` to `1.10`). This is in contrast to in-

place upgrades where it is required to upgrade to each intermediate patch release.

Control plane

To install a new revision called `canary`, you would set the `revision` field as follows:

In a production environment, a better

revision name would correspond to the Istio version. However, you must replace `.` characters in the revision name, for example, `revision=1-6-8` for Istio `1.6.8`, because `.` is not a valid revision name character.

```
$ istioctl install --set revision=canary
```

After running the command, you will have two control plane deployments and services running side-by-side:

```
$ kubectl get pods -n istio-system -l app=istiod
NAME                                    READY   STATUS    RESTAR
TS   AGE
istiod-786779888b-p9s5n                 1/1     Running   0
     114m
istiod-canary-6956db645c-vwhsk          1/1     Running   0
     1m
```

```
$ kubectl get svc -n istio-system -l app=istiod
NAME            TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)
                                                AGE
istiod          ClusterIP   10.32.5.247   <none>        15010/TC
P,15012/TCP,443/TCP,15014/TCP                   33d
istiod-canary   ClusterIP   10.32.6.58    <none>        15010/TC
P,15012/TCP,443/TCP,15014/TCP,53/UDP,853/TCP   12m
```

You will also see that there are two sidecar injector configurations including the new revision.

```
$ kubectl get mutatingwebhookconfigurations
NAME                             WEBHOOKS   AGE
istio-sidecar-injector           1          7m56s
istio-sidecar-injector-canary    1          3m18s
```

Data plane

Unlike istiod, Istio gateways do not run revision-specific instances, but are instead in-place upgraded to use the new control plane revision. You can verify that the `istio-ingress` gateway is using the `canary` revision by running the following command:

```
$ istioctl proxy-status | grep $(kubectl -n istio-system get pod
 -l app=istio-ingressgateway -o jsonpath='{.items..metadata.name
}') | awk '{print $7}'
istiod-canary-6956db645c-vwhsk
```

However, simply installing the new revision has no impact on the existing sidecar proxies. To upgrade these, you must configure them to point to the new `istiod-canary` control plane. This is controlled during sidecar injection based on the namespace label `istio.io/rev`.

To upgrade the namespace `test-ns`, remove the `istio-injection` label, and add the `istio.io/rev` label to point to the `canary` revision. The `istio-injection` label must

be removed because it takes precedence over the `istio.io/rev` label for backward compatibility.

```
$ kubectl label namespace test-ns istio-injection- istio.io/rev=
canary
```

After the namespace updates, you need to restart the pods to trigger re-injection. One way to do this is using:

```
$ kubectl rollout restart deployment -n test-ns
```

When the pods are re-injected, they will be configured to point to the `istiod-canary` control plane.

You can verify this by looking at the pod labels.

For example, the following command will show all the pods using the `canary` revision:

```
$ kubectl get pods -n test-ns -l istio.io/rev=canary
```

To verify that the new pods in the `test-ns` namespace are using the `istiod-canary` service corresponding to the `canary` revision, select one newly created pod and use the `pod_name` in the following command:

```
$ istioctl proxy-status | grep ${pod_name} | awk '{print $7}'
istiod-canary-6956db645c-vwhsk
```

The output confirms that the pod is using `istiod-canary` revision of the control plane.

Stable revision labels (experimental)

If you're using Helm, refer to the Helm upgrade documentation.

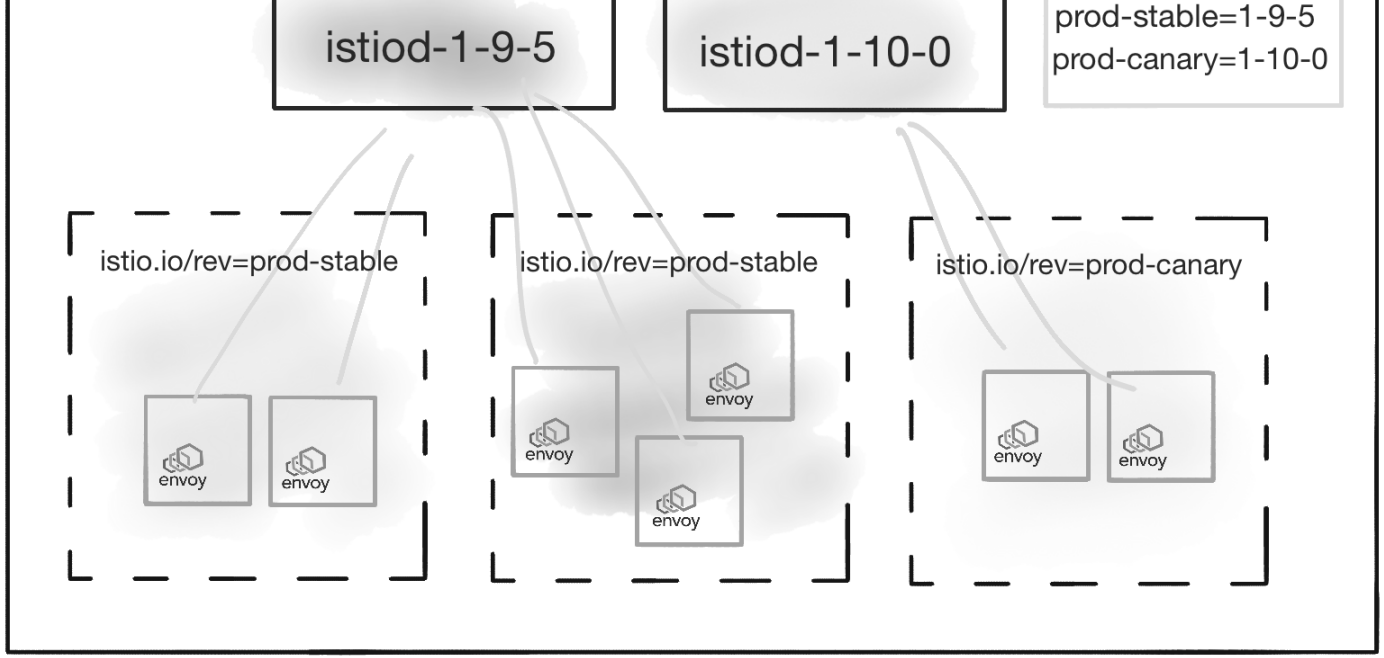
Manually relabeling namespaces when moving them to a new revision can be tedious and error-prone. Revision tags solve this problem. Revision tags are stable identifiers that point to revisions and can be used to avoid relabeling namespaces. Rather than relabeling the namespace, a mesh operator can simply change the tag to point to a new revision. All namespaces labeled with that tag will be updated at the same time.

Usage

Consider a cluster with two revisions installed, `1-9-5` and `1-10-0`. The cluster operator creates a revision tag `prod-stable`, pointed at the older, stable `1-9-5` version, and a revision tag `prod-canary` pointed at the newer `1-10-0` revision. That state could be reached via these commands:

```
$ istioctl tag set prod-stable --revision 1-9-5
$ istioctl tag set prod-canary --revision 1-10-0
```

The resulting mapping between revisions, tags, and namespaces is as shown below:



Two namespaces pointed to prod-stable and one pointed to prod-canary

The cluster operator can view this mapping in addition to tagged namespaces through the `istioctl`

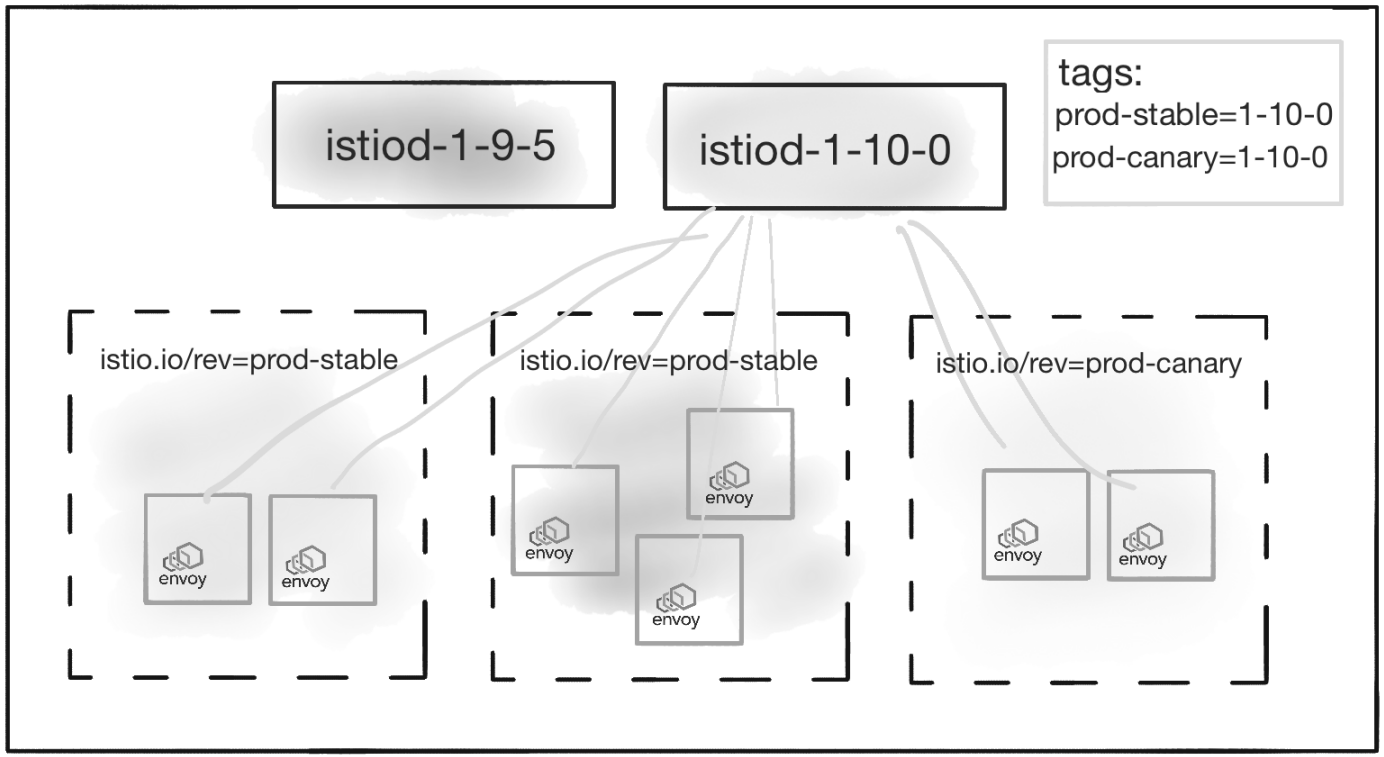
`tag list` command:

```
$ istioctl tag list
TAG         REVISION NAMESPACES
prod-canary 1-10-0   ...
prod-stable 1-9-5    ...
```

After the cluster operator is satisfied with the stability of the control plane tagged with `prod-canary`, namespaces labeled `istio.io/rev=prod-stable` can be updated with one action by modifying the `prod-stable` revision tag to point to the newer `1-10-0` revision.

```
$ istioctl tag set prod-stable --revision 1-10-0
```

Now, the situation is as below:



Namespace labels unchanged but now all namespaces pointed to 1-10-0

Restarting injected workloads in the namespaces marked `prod-stable` will now result in those workloads using the `1-10-0` control plane. Notice that no namespace relabeling was required to migrate workloads to the new revision.

Default tag

The revision pointed to by the tag `default` is considered the ***default revision*** and has additional semantic meaning.

The `default` revision will inject sidecars for the `istio-injection=enabled` namespace selector and `sidecar.istio.io/inject=true` object selector in addition to the `istio.io/rev=default` selectors. This makes it possible to migrate from using non-revisioned Istio to using a revision entirely without relabeling namespaces. To make a revision `1-10-0` the default, run:

```
$ istioctl tag set default --revision 1-10-0
```

When using the `default` tag alongside an existing non-revisioned Istio installation it is recommended to remove the old `MutatingWebhookConfiguration` (typically

called `istio-sidecar-injector`) to avoid having both the older and newer control planes attempt injection.

Uninstall old control plane

After upgrading both the control plane and data plane, you can uninstall the old control plane. For example, the following command uninstalls a control plane of revision `1-6-5`:

```
$ istioctl x uninstall --revision 1-6-5
```

If the old control plane does not have a revision label, uninstall it using its original installation options, for example:

```
$ istioctl x uninstall -f manifests/profiles/default.yaml
```

Confirm that the old control plane has been removed and only the new one still exists in the cluster:

```
$ kubectl get pods -n istio-system -l app=istiod
NAME                            READY   STATUS    RESTARTS   AG
E
istiod-canary-55887f699c-t8bh8   1/1     Running   0          27
m
```

Note that the above instructions only removed the resources for the specified control plane revision, but not cluster-scoped resources shared with other control planes. To uninstall Istio completely, refer to the uninstall guide.

Uninstall canary control plane

If you decide to rollback to the old control plane, instead of completing the canary upgrade, you can

uninstall the canary revision using `istioctl x uninstall --revision=canary`.

However, in this case you must first reinstall the gateway(s) for the previous revision manually, because the uninstall command will not automatically revert the previously in-place upgraded ones.

Make sure to use the `istioctl` version corresponding to the old control plane to reinstall the old gateways and, to avoid downtime, make sure the old gateways are up and running before proceeding with the

canary uninstall.